Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.2302.07

2023, 42(2) 54-70

An improved non-local awareness of congestion and load balanced algorithm for the communication of on chip 2D mesh-based network

Munib Ahmed *, Muhammad Iram Baig

Department of Electrical Engineering, University of Engineering and Technology, Taxila 47050 Pakistan

* Corresponding author: Munib Ahmed, Email: munibahmed340@gmail.com

Received: 27 July 2022, Accepted: 24 March 2023, Published: 01 April 2023

KEYWORDS

Congestion Awareness

Load Balancing

Network-on-Chip

Network Partition

Source Routing

Distributed Routing

ABSTRACT

Due to advancements in multi-core design technology, IC (Integrated Circuits) designers have expanded the single chip multi-core design. A privileged way of communication effectively between these multi-cores is a Network on-chip (NoC). Design of an effective routing algorithm capable of routing data to non-congested paths is the most notable research challenge in NoC, by retrieving congestion information of non-local nodes. This research proposed an improved congestion-aware load balancing routing algorithm. Non-local or distant links congestion awareness is done by propagating congestion information via data packets. By counting number of hops from the source node, in the quadrant of the destination node, an intermediate node has been defined, and after the calculation of the least congested route to the intermediate node, this route is also stored in the data packet for source routing. Furthermore, for load balancing network is partitioned into two areas called high congested area (HCA) and low congested area (LCA). For load balancing, from HCA a node in LCA is selected as output for data packets. Comparison of the proposed algorithm is done in the form of average latency, average throughput, power consumption, and scalability analysis under synthetic traffic patterns. Under simulation experiments, it is shown improvement in an average latency and throughput of the proposed algorithm is 31.28% and 5.28% respectively, than existing.

1. Introduction

In the current era, with the consideration of the high amount of data generated, multi-core processors and single Core processors could not process massive data [1]. So, number of multi-core processor required to process massive data. Increase in the number of cores on a single chip, communication between these cores could not be fulfilled by bus-based or point-to-point architecture. In this regard, a Network-On-Chip (NoC) is a broad field under research, an alternative to these architectures for scalability and efficiency improvement [2]. Unlike the bus-based and point-to-point architecture, NoC has the benefit of using independent implementation and optimization of nodes, multiple topologies support, and network link and nodes customization as per application. Traditional interconnect difficulties have been overcome with the use of customized links between each node because it employs a fast and efficient link to route packets between routers. With better utilization of resources, NoC also provides high bandwidth and throughput. NoC has a scale-able architecture, as a result debugging and expanding the network is easier [3].

According to International Technology Roadmap for semi-conductor (ITRS) report, the number of

processing cores would be more than five thousand by 2025 [4]. With this road map in semiconductor technology, the new requirement must be merged for NoC. The new requirement, such as Routing algorithm for traffic mapping, load balancing, and congestion management. This increase in the number of cores on the Integrated Circuit is a serious issue for the researcher to efficiently. In NoC, communication between each core (also called nodes) is done by routing unit called Processing Element (PE) via data link. This PE is programable and is configured for routing the data packet control. In this paper, we discuss load balancing and congestion awareness by proposing an improved congestion-aware load balancing algorithm.

In a network, congestion occurs when a link exceeds its capacity of data. The average distance between the source and destination nodes grows as the network size increase[5], while the number of cores raises the probability of congestion. Corresponding nodes with these links are said to be a congested node. A congested node holds packets since resource/nodes are occupied by packets allowing incoming packets not to route to their destination. Therefore, congestion in the network cause packet not to reach a destination in time and hence the degraded important network parameter i.e., throughput and latency. In computer networks, when congestion occurs packets are discarded. In NoC packets could not be discarded because NoC a lossless packet routing is required. As a result, there is a possibility of high-power consumption as well as a sharp efficiency degradation [6].

To improve the NOC performance many approaches have been proposed, including changing packet injection rate dynamically [7] [8], optimizing routing algorithm [9] [10], increasing the number of router virtual channels (VCs), and changing NOC structure, and so on. Designing a routing algorithm or optimization [11, 12] is one of the best effective solutions. Which specifies the path of a packet to be transmitted on it and thus it directly influences the latency and throughput of NoC [10, 13]. In various ways, congestion in the network could be controlled by using a congestion-aware routing algorithm.

In a congestion-aware routing algorithm, a path is selected having the least congestion by knowing congestion information of non-local(distant) routers. In general, routing algorithms are divided into deterministic, oblivious, and adaptive algorithms. Deterministic routing algorithm: just one path is used to send packets between two particular nodes. Oblivious routing: a random set of nodes is chosen from the source node, with no regard for the network state. Adaptive routine algorithm: packet path is selected depending on the network state. A less congested path would be selected out of multiple congested paths. Adaptive Routing algorithms are subdivided into two groups, A minimal adaptive routing method selects the route that requires the fewest number of hops. Non-minimal adaptive routing: An adaptive routing algorithm finds a route that increases the hops count from source to destination by increasing the degree of adaptivity. Minimal routing techniques increase latency and power consumption since packets are routed via a large number of links and routers. Non-minimal routing algorithms, on the other hand, are best for choosing a non-minimal path with load balancing that keeps packets out of congested areas, reduces packet latency, and allows them to reach their destination [14].

Adaptive routing algorithms are separated into two types based on congestion awareness: local congestion awareness and non-local congestion awareness. For congestion awareness, different techniques are proposed in the literature. In recent studies congestion propagation network (CPN)[15] which employs additional hardware for congestion propagation, reinforcement learning based method [16] utilizes the back aunt method for congestion awareness of upstream router and increase the traffic burden in network with zero area overhead, link sharing [17] to propagate the congestion information on Idea link with the disadvantage of finding and forcing ideal clock cycle for congestion propagation and freerider [18] to the propagation of congestion information via data packet. Starvation may occur when the packet does not reach in time. In our research propagation is done by using freeride without area overhead of additional network and traffic burden for congestion carrier. Classification of a routing algorithm can also be done as Source Routed (SR) algorithm and Distributed Routed (DR) algorithm. In source routing a precomputed route is stored in a packet. Whereas in DR the packet holds only the address of the destination node. In our research, SR and DR with minimal adaptive routing is utilized.

The advantage of using a minimal routing Algorithm is packet does not go far away from the destination node and hence avoids live-lock in the network. While using a minimal routing algorithm there is a problem arises called load balancing all over the network. ZARA [15] and ParRouting [19] partition the network, in [20] network is segmented into subnets for load balancing in the network. In our research load

balancing is done by partitioning the network into two areas based on closeness centrality.

The main contribution of this research is as flows:

- An intermediate node is defined in the quadrant of the destination node from the source node, and the least congested path is calculated for source routing.
- For load balancing network is partitioned into two area HCA and LCA, route of data packet from one area to other is take placed based on proposed algorithm.
- Improved congestion awareness routing algorithms for both LCA and HCA are proposed using source routing and distributed routing by leveraging congestion propagation via data packet, avoiding the separate congestion propagation network, or wiring overhead for propagating the congestion information.

The rest of this paper is provided as follows:

Section 2 summarizes the relevant work, and section 3 gives the proposed algorithm. The implementation of the proposed algorithm is shown in Section 4. Experimental setup and performance analysis are given in Section 5. Finally, in section 6, a conclusion has been drawn.

2. Literature Review

Routing algorithms with the feature of congestion awareness of distant and only neighbor nodes proposed so far in the literature are covered in this section. Any routing algorithm having the property of congestion awareness first defines the output nodes (based on congestion status) to deliver the packet to the destination. Based on the crowd information, an appropriate node is picked among the output nodes, and the packet is transmitted to the destination through this node. An adaptive routing method that identifies whether the output node is crowded is known as a congestion-aware routing algorithm. Because of its simplicity in NoC design, the non-adaptive scheme is very well-known. This type of routing is XY-routing, which is a non-adaptive routing scheme [21]. In this algorithm, packets are first routed to a horizontal plane and then routed to a vertical plane to reach the destination node. This algorithm routes packets via the shortest route. On the other hand, adaptive routing algorithms dynamically adapt the route according to the condition of the network and are mostly used for load balancing and congestion awareness.in adaptive routing algorithms, virtual or physical channels are included to ensure the dead lock freedom in the routing algorithms.

The congestion measurement determines whether the node is crowded or not. The congestion measure is a parameter that determines the status of the node whether it is congested or not. The quantity of full or empty buffer [22], number of VCs full or empty [23], and count of arbiter requests [24] are used to calculate this value. A link to the next node is set to be congested if credit remains for buffer are less than a certain threshold. Similarly, for VCs, the quantity of free virtual channels determines the congestion status. The number of flits that pass through any link is also specified by a crossbar. The larger the number of flits passes which indicates the node is congested. In [25] measure of this parameter is done by two-bit values. Each value defines four distant thresholds and defines a different level of congestion based on the buffer level.

Another essential parameter in a congestion-aware routing algorithm is the number of nodes required to find the next node using congestion information from these nodes. The algorithm is classified into three types based on this criterion: local, regional, and global congestion algorithms. Only neighboring nodes determine congestion information in a local congestion-aware algorithm. Hence the propagation extent is limited to only neighbor nodes. In Dy-Adaptive [25] and Dy-XY [22] each node has congestion information of only the neighbor node to decide the routing of a packet to a later node. For example, in Dy-Adaptive routing algorithm, each node determines the congestion information neighbor node and selects the candidate node having the least congested link. The link congestion is determined by the buffer level of a router.

When both output links are in minimum route, the BARP [26] in 2008 algorithm is used, in which the packet is sent to the next node estimate the probability computed by neighbor congestion in a grouping. Tedesco et al. [27] presented a new routing strategy in which a packet called an alarm is sent to the source node informing it of the intermediate node's congestion state. The congestion information is stored in a table and routes the package by using adaptive source routing. Selecting the route based on the local congestion or neighbor congestion information may lead to the wrong route because congestion might exist in distant nodes. For local congestion, there was still no new research on congestion-aware routing algorithms.

The regional congestion aware algorithm is another sort of congestion-aware routing algorithm that

propagates congestion information from a few far-flung nodes. This algorithm, once again, does not provide congestion information for all nodes in the network. Gratz et al. presented RCA (Region Congestion Awareness) [24] in 2008, which was the first regional congestion-aware algorithm. This research proposed three methods of congestion propagation RCAID, RCA Fan-in, and RCA Quadrant. A simple implementation is RCAID which propagates congestion information autonomously along each dimension. In RCA Fan-in, congested information in an axis along with congested information of orthogonal nodes are also added. As a result, a current router has more congestion information on the network. Because of gathering the wrong congestion information from the orthogonal direction third method, the RCA quadrant comes into existence. In RCA Quadrant, two different values of two quadrants were received, updated, and propagated in the network with the disadvantages of wires and logic complexity over RCA Fanin and RCAID. The RCA routing algorithms suffer from two defects. The first defect is it overlooks the destination location to collect congestion information. For example, if the distance between a source node and destination node is less than five hops, interference may occur with the congestion information of nodes located beyond the destination nodes. The second problem is that the RCA collects the congestion information of nodes along the X-axis and Y-axis only. Knowing the fact that most of the routes from the source node do not cross these notes. To solve this issue DBAR [28] and CATRA [29] were proposed after the RCA algorithm.

In DBAR, the routing algorithm used two congestion information registers for nodes congestion information in rows and columns. In this way, the interference that may occur in RCA is resolved by DBAR by determining the congestion information from registers following source node and destination node. But still, DBAR has only congestion information of rows and columns. The CATRA algorithm, which is one of the most prominent in modern years, is based on the chance of passing through any intermediary node. The congestion details of trapezoid nodes around the present node are being gathered and compared in the CATRA algorithm. Agents are utilized to spread the congestion condition. The drawback of CATRA is that the agent is overburdened. In 2015, a freerider [18] method was presented to solve this problem. Instead of using a specialized Congestion Propagation Network, congestion bits are spread via data packets in freerider. The Overhead of agents used in CATRA is reduced by this method. According to the author out of 128bits flit, 52 bits are free and could be used for propagation of the congestion information of distant or non-local nodes.

In 2017, a new method called ERCA [30] was created to solve the problem of RCA interference. For congestion estimate, ERCA uses dynamic weight rather than static weight. The distance between the source and destination nodes is used to calculate these weights. AZRA [15], a novel load balancing method, was suggested in 2018. A forbidden region was established in this method, and packets with certain attributes were not permitted to transit through it. In 2020, ParRouting [19] algorithm for load balancing was proposed. In this algorithm, a network is partitioned into three areas based on closeness centrality and packet from the low priority area, and the highly congested area is routed to the high priority area and low congestion area. The disadvantage of an algorithm is that at a high injection rate the packet moves toward high priority nodes get congested and degrade in latency if the destination node lies in the central node. In 2020, Akbar et al [31] proposed an algorithm that is a combination of both deterministic and adaptive routing. In extreme congestion, the packet moves as far as possible in a limited time and again selects the minimum route by knowing the possibility of congestion calculated by using betweenness centrality, history of previous packets routed, and degree of adaptivity. DAR [32]and GCA [33] are global congestion-aware routing algorithms. The disadvantage of these algorithms is extra overhead for storing and propagating global congestion information and is not scalable. Because of this most global congestion-aware algorithms are not proposed in recent years. However, a network is segmented into a subnet and then a global congestion-aware algorithm is implemented on them, [20] is one of these categories of an algorithm. NoC was divided into numerous subnets in this routing technique. The proposed method coupled the benefits of reginal congestion-aware routing in each subnet with global congestion-aware routing between subnets. CPN agents require more space and power in regional and global algorithms, making them uneconomical for the NoC design used in battery-powered or portable embedded systems. That is why, in recent years, there has been a minimal study on this type of routing algorithm. The employment of a regional congestion algorithm is projected to be no longer cost-effective due to the relevance of area and resources.

As a result, a minimum adaptive congestion aware load balance with congestion information from a few non-local nodes up to three hops from the source node is presented in this study. In the following, it can be

seen how intelligently a least congested path to an intermediate node is selected, and other decisions for route selection are made based on the residence of the destination and source node in the network. Table 1 illustrate the summary of Cutting edge research papers using different routing technique and performance parameter measuring.

Table 1

Summary of Cutting-edge research papers having deficiencies, routing technique and performance parameter measuring

Sr#		year	Routing Type	Routing Characteristic/Deficiencies	Switching Type	Performance Parameter
1	Nain Z.et al [36]	2020	Adaptive +Fault Tolerance	Live lock avoidance by selecting alternative port Deficiencies: No distant or non-local congestion awareness	wormhole	Avg. Flit Delivery ratio 0.42flits/cycle/node @ 0.03 IR
2	Akbar R et.al [31]	2020	Deterministic adaptive Routing	Congestion-aware probabilistic method based.	wormhole	Normalized power, consumption, Avg. Latency, Throughput 74 cycles @ 0.01 IR bit reversal traffic Patterns
3	Subnet-Routing [20]	2021	Fully Adaptive Routing	Congestion aware by use of advantage of both local and global congestion awareness Deficiencies: overhead of using separate CPN	wormhole	Latency, Throughput power, switching activity scaling analysis. 40.05 cycles @ 0.01 IR with synthetic Bit Reversal traffic
4	Par-routing [19]	2020	Adaptive	Partitioning network and set priorities for local balancing Deficiencies: if destination node in central area and packet in high priority area, the is degradation in latency and throughput	wormhole	Avg Latency, Throughput, Switching activity 62 cycles @ 0.01 injection rate of bit reversal traffic
5	Ahmad K et al [37]	2021	Adaptive	Congestion Aware Deficiencies: congestion awareness via data packet on for one hop beyond the neighbor node	wormhole	Latency

3. Proposed Algorithm Design

This section elaborates on the design of the congestion-aware load balancing routing algorithm. The proposed algorithm is a type of destination-based Minimal Adaptive using both SR and DR. The core idea is to define an Intermediate node, three hops beyond the source node either in the x-axis direction or in the y-axis direction. The selection of the intermediate node is described below. Once the Intermediate node has been defined, the least congested path from the source node is selected for source routing. This section also describes the partition of a network in HCA and LCA for load balancing. For both congested areas two Algorithm 1 and Algorithm 2 are proposed. Congestion propagation and updating mechanism are described in section 4.

3.1 Intermediate Node Selection

An intermediate node is a node that is either in the x-axis direction or in the y-axis direction from a source node. Selection is made based on a minimum value of the congested node either on the x-axis or y-axis. This is illustrated in Fig.1. This intermediate node could be in any direction from source to destination in the quadrant North-East (NE), East-South (ES), North-West (NW) or West-South (WS) depending upon the destination node. In Fig.1 the selection of intermediate node I_x and intermediate node I_y depends upon the

congestion of the link along the x-direction and y-direction respectively.

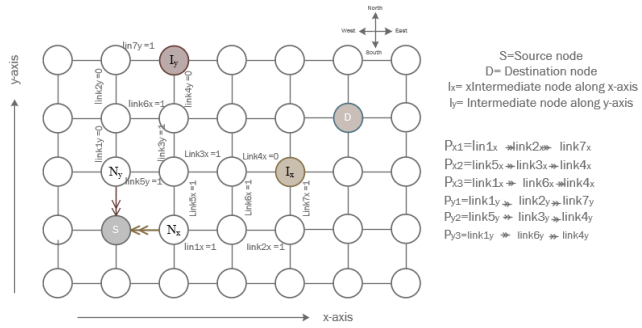


Fig. 1. Intermediate node I_x and I_y are determined in the NE quadrant of destination node D and I_y selected based on the least congested path P_{y3}

An Intermediate node is a predefined node calculated from the location of the source node and destination node quadrant. If source node having location of $S_{(x,y)}$, intermediate node along x-axis is calculated as $I_x=I_{(x-1,y+3)}$ for NE quadrant, $I_x=I_{(x+1,y+3)}$ for ES quadrant, $I_x=I_{(x-1,y-3)}$ for NW quadrant, $I_x=I_{(x+1,y-3)}$ for WS quadrant nearer to destination node and along y-axis is calculated as $I_y=I_{(x-3,y+1)}$ for NE quadrant , $I_y=I_{(x+3,y+1)}$ for ES , $I_y=I_{(x-2,y-1)}$ for NW and $I_y=I_{(x+3,y-1)}$ for WS quadrant.

All the congestion information of links along the x-axis and y-axis is stored in the congestion information register module in Router (further detail in router architecture subsection). A link having value of 1 that represents the link is congested and 0 means the link is not congested. Notation of congestion scale depends upon the free cell in the buffer of the router (buffer level). Which is given in Table 2. In [20] different buffer levels occupied are presented by 2-bit value for congestion representation. Same congestion measure value we are using here. If the congestion information value is binary value 11 or more than 75% link is said to be congested and marked as 1 for link congestion propagation.

Table 2
Amount of congestion determined for input buffer:

Input buffer level occupied in percentage	Congestion Value
Less than or equal to 25%	1
In between 25% and 50%	2
Between 50% and 75%	3
More than 75%	4

For intermediate node selection, out of three paths P_{x1} P_{x2}, and P_{x3} along x- the axis and along the y-axis P_{y1}, P_{y2}, and P_{y3} least congested path is selected. For example, in Fig.1 P_{x2} and P_{y3} are the least congested path along the x-axis and y-axis respectively. After comparing the congestion link of selected paths P_{x2} and P_{y3} a link having the least congested path defines the intermediate node. P_{x2} has a link value of 110 and P_{y3} has a link value of 100. P_{y3} defines the selected intermediate node along the y-axis I_y. So, a packet is routed along the y-direction (north) using source routing. If both paths have equal congestion links, the packet is routed to the intermediate node nearer to a destination node. If the destination node is one or two hops (less than three hops) beyond the source node, then local congestion information is utilized for output link selection.

3.2 Load Balancing

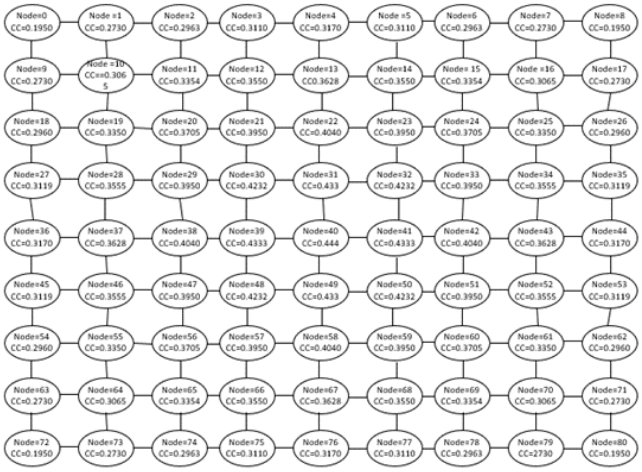
Adaptive congestion-aware routing algorithm is a load balance routing algorithm. While using a minimal routing algorithm there may occur a problem of load balancing. A step must be taken to distribute traffic load all over the network. ParRouting [19] partitioned the network into three regions to balance the heavy load traffic from the central area to the Edge area. The same technique we are using here to partition the network by using closeness certainty. The shortest pathways from node μ to all nodes are measured by closeness centrality. It's commonly written as the normalized inverse of the sum of the graph's topological distances. Mathematically it is written as:

$$C(\mu) = \frac{n-1}{\sum_{v}^{n-1} d(\mu,v)} \quad (1)$$

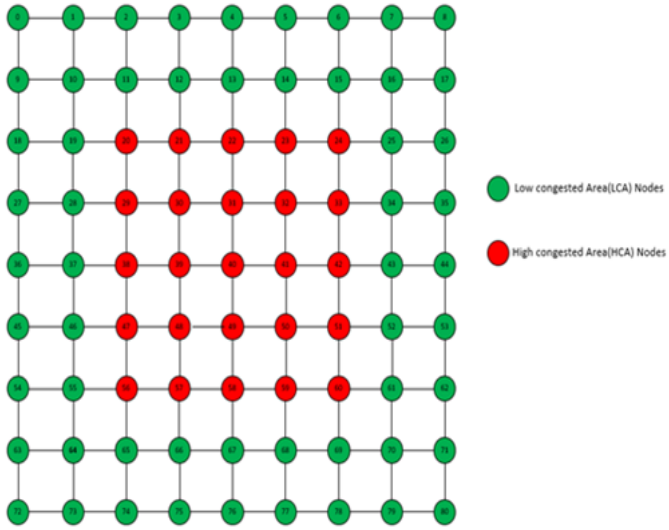
In above Eq.1 shortest distance of path between *v* and u is d (*v*, μ), and n is the total number of nodes that could be accessed by μ in the mesh network. A network is partitioned into two areas HCA and LCA. Based on two partitioned area Algorithm 1 and Algorithm 2 is proposed. Output link for HCA is selected toward LCA to balance traffic from the highly congested area to the low congested area. An example of partition 9x9 2D mesh is shown in Fig. 2(a). It can be seen that *Max (C(μ))* is 0.444 and *Min(C(μ))* is 0.1950. Patriation of the network is on based on closeness centrality and every node whose closeness centrality is greater than some threshold is considered as HCA. Calculation of threshold is as flows.

$$Threshold = K\,[\,Max\,(C(\mu)) - Min(C(\mu))\,] + Min(C(\mu)),\ where\quad 0<K<1 \quad (2)$$

The value of K is range from 0 to 1. If we increase the value of K, HCA will be smaller. If we decrease the value of K, HCA would be larger and load balancing would not be possible. Here we choose the value of K=0.7.



(a)



(b)

Fig. 2. (a) 9x9 2D mesh Network with closeness certainty value. (b) The network is partitioned into two are HCA and LCA based on a threshold value.

All values of closeness centrality for nodes greater than this threshold are considered HCA. In Fig.2 (b) all red nodes are HCA and green nodes are LCA. If the source and destination node are in HCA, then Algorithm 1 is applied. If source and destination both or only destination node (no matter source node in HCA) are in LCA, algorithm 2 is applied.

Algorithm 1

For High Congestion Area (HCA) Nodes

```
void x_direction _path_computation()
 Begin
     Px= encoded_min_path(Px1, Px2 , Px3);
     Call fillrouteInHeader( packet_header);
 end
```

```
void y_direction _path_computation()
     Begin
      Py= encoded_min_path(Py1, Py2 , Py3);
     Call fillrouteInHeader( packet_header);
     end
```

```
Step1:
Begin
      if destination node is next to source node then
        route packet to the neighbor node (Nx or Ny) without checking congestion status;
      end if
      else if buffer level of Nx or Ny is less than 25% then
            if Buffer level of Nx and Ny equal then
                  Route packet to neighbor node nearer to destination;
                  RouteIsSet = false;
            end if
             else compare buffer level of Nx and Ny
                  Route packet to neighbor having least buffer level;
            end if
 end

            if  min_CongestionPath_X <  min_CongestionPath_Y  then
                  X_direction_path_computation ();
                  RoutIsSet=true;
                  NoH=4;
            end if
            else if min_CongestionPath_X >  min_CongestionPath_Y  then
                  Y_direction_path_computation ();
                   RoutIsSet=true;
                   NoH=4;
            end if
            else if  min_CongestionPath_X == min_CongestionPath_X     then
            Route to the path having intermediate node (Ix or Iy) nearer to destination;
                  RoutIsSet=true;
                  NoH=4;
             else select randomly intermediate node and compute path to node;
            end if
      end if
 end if
```

Step2:

```
 Begin
if RouteIsSet == true  then
      goto step3;
     end if
     else if destination node (dst) lies outside the intermediate node Ix and Iy  and Ix and Iy
location not outside network dimension  then
           if NE-Quadrant according to destination node          then
              #only for NE-quadrant path computation is discussed here
                  Px1=distx [0][0] +distx [0][1] + distx [0][6];
                  Px2= distx [0][0] +distx [0][5] + distx [0][3];
                  Px3= distx [0][4] +distx [0][2] + distx [0][3];
                  Py1=disty [1][0] +disty [1][1] + disty [1][6];
                  Py2= disty [1][0] +disty [1][5] + disty [1][3];
                  Py3= disty [1][4] +disty [1][2] + disty [1][3];
                  min_CongestionPath_X =compare (Px1, Px2, Px3);
                  min_CongestionPath_Y =compare (Py1, Py2, Py3);
```

Step3:

```
      Begin
            Set the next hop according LSB in header;
            Shift left the address bit for next hop;
            NoH --;
            If  NoH ==0   than
            RouteIsSet =false;
            end if
      end
```

3.3 Algorithm 1 Strategy

The network is partitioned into two areas, HCA and LCA. Algorithm 1 applies to HCA if the source node and intermediate nodes (both I_x and I_y) are in HCA. A network has a heavy load in its central area [19]. It is necessary to route a packet from this congested area to a low congested area. This is true for Non-Minimal Adaptive routing. For Minimal Adaptive routing, the packet must be a queue at the input of the buffer. Queueing packets at input of Router buffer increases queue latency and hence causes performance degradation. Algorithm 1 gives a solution by choosing the least congested path with minimal Adaptive Routing.

Assume that destination (dst) node lies in the NE quadrant. In the first Step, Algorithm 1 checks the buffer level of neighbor nodes N_x and N_y. If the Buffer scale of x neighbor N_x(East) and y neighbor node N_y(North) is less than or equal to 25%(See Table 2), the packet will be routed to the neighbor node nearer to the destination node.

If the first step is not executed, it enters the second step. In this step, it first checks *RoutIsSet*(further detail in section 4 of packet structure) from the packet header. If *RoutIsSet* is true it will enter in step3 and perform source routing. Otherwise, it checks *dst* nodes lie outside the region of Intermediate node I_x and I_y, an Intermediate node with the least congested route is selected along x- the direction or along the y-direction. Three paths from the source node along x-direction P_{x1}, $P_{x2,}$ and P_{x3} are compared with three paths P_{y1}, $P_{y2,}$ and P_{y3}. Each path has three hops to reach the intermediate node. These hops contain a link between two nods, a link is said to be congested if it has a value of 1. Non-congested link marked as 0. Each link is represented by distant link(dist[0][i] for x-direction and dist[1][i] for y-direction path to intermediate node) that and consume only one bit in packet header for congestion information. All distant links corresponding to the path are shown in Fig.3. Criteria for the congested link or non-congested link are based on the buffer scale in the router as described above. After Least congested path calculation the encoded complete path is loaded to the packet header. In the algorithm, remote link addresses are only for the EN quadrant. The other three remaining quadrant path computation is done in the same way. *encoded_min_path* function performs encoding of a complete path in three bits. It first compares three paths, if two paths or all three paths have the same congestion value, the address of the randomly chosen path is a return to P_y or P_x. Table 2 shows a total number of six paths along x-direction and y-direction are encoded with 4bits. 0 and 1 represent the output node along the x-axis in the destination node direction and the output node along the y-axis in the direction of the destination node respectively. If the destination node lies in the region of the Intermediate node near to source node, the buffer level of output neighbor N_x or N_y is checked. If it is less than or equal to 2(link congestion between 50% to 70%), the output node having the least congested link or nearer to the destination is selected. If congestion of neighbor is greater 2(more than 70%) output node having low closeness centrality is selected.

In Step 3, the Number of Hops (NoH) is equal to 3, the maximum number of hops to reach the intermediate node. Whenever a packet enters the source node, if *RoutIsSet* is true, it is decremented until it reaches zero then *RouteIsSet* is set as false.

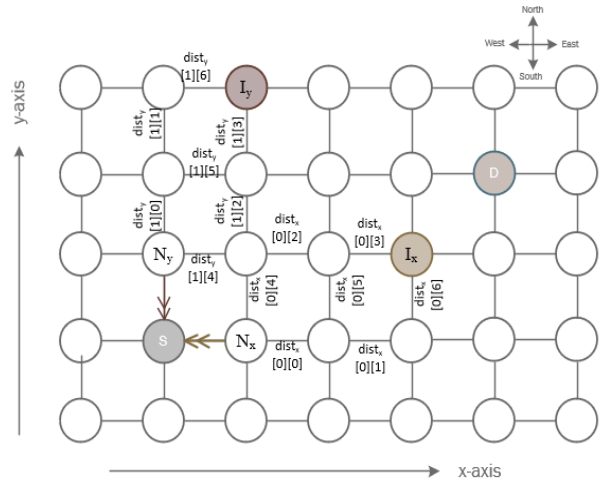


Fig. 3. Remote distant links represent congestion information

At the same time, the least significant bit (LSB) of the encoded path is left shifted to determine the next direction for the source node.

Table 3

Addresses of paths encoded along the x-direction and y-direction:

Paths along x- axis and y-axis	Encoded address in binary
P_{x1}	0001
P_{x2}	0100
P_{x3}	0010
P_{y1}	1110
P_{y2}	1011
P_{y3}	1101

Algorithm 2

For Low Congestion Area (LCA) Nodes

```
Begin
    if  closeness centrality of source node and destination node is less than threshold
        and neighbor node is not an edge node  then
// for NE-Quadrant, another quadrant not given
      if  dist_x[0][0]+dist_x[0][1]+ dist_x[0][5] + dist_x[0][6]) < dist_y[1][0]+dist_y[1][1]+
          dist_y[1][5] + dist_y[1][6])  then
        route packet to neighbor along x- direction (N_x) ;
      end if
      else if  dist_x[0][0]+dist_x[0][1]+ dist_x[0][5] + dist_x[0][6])> dist_y[1][0]+dist_y[1][1]
            dist_y[1][5] + dist_y[1][6])
            route packet to neighbor along y- direction (N_y) ;
    end if
    else  route packet to neighbor N_x or N_y nearer to destination node;
  end if
  else if  closeness centrality of source node is greater than threshold and destinatio
          node closeness centrality greater than threshold  then
        if  closeness centrality of intermediate node I_x and I_y greater than threshold
        then
            Goto algorithm1;
        end if
        else route the packet to low closeness certainty Neighbor node;
   else  check buffer level of neighbor (N_x and N_y)
        route packet to least congested neighbor link;
      if  buffer level of neighbors is same  then
        Route packet to neighbor nearest to destination;
      end if
end
```

3.4 Algorithm 2 Strategy

Algorithm 2 applies to nodes that are in LCA. All nodes having closeness centrality less than the threshold from Eq. 2 are said to be in LCA. At first, this algorithm checks whether the source node and destination node are in LCA and checks whether any of the Neighbor nodes is in the boundary node or not. If the source node and destination are both in LCA and not a boundary node, the neighbor node N_x or N_y is selected in the direction of the destination node by calculating distant congestion information. Distant link congestion up to 3-backyard along with x-axis nodes and y-axis nodes are compared. The neighbor candidate node (N_x or N_y) having the least congestion is selected as the output node. If the comparison of link congestion is equal, a neighbor node with the nearest destination is selected. If the residence of the source node is in the HCA and the destination node in LCA, the location of both intermediate nodes I_x and I_y is checked. if I_x and I_y both are in HCA, Algorithm 1 is applied. Otherwise, the packet from High congestion is routed to LCA based on no closeness centrality.

4. Implementation of Algorithm

4.1 Congestion Propagation and Calculation

Congestion-aware routing algorithm requires no-local congestion information. Unlike congestion propagation network (CPN) proposed in CATRA to propagate real-time congestion information through a dedicated network adds extra area overhead for network wiring, updating and propagation of congestion information being done via data packets. According to Chen et.al [18] , in 128bits of NOC, the head flit of the packet contains more than 52 free bits as shown in Fig**.** 4 these free bits can be used for the propagation of congestion information.

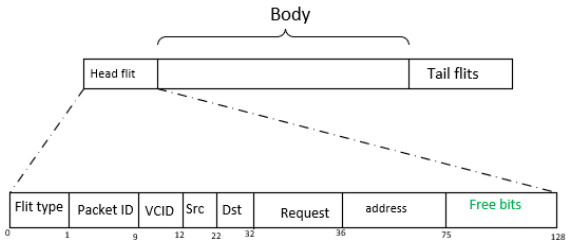


Fig. 4. Packet structure for 128-bit mesh based network [18]

The core idea is that head flit coming from North, south, east, and west have congestion status of K-backyard link [18]. Local Congestion information of four links is updated when head flit enters the router. So, when the head flit enters the router, at first congestion information of four local links (North, South, East, and West) is calculated based on the buffer level of the downstream and upstream routers. Secondly, all congestion information of K-backyard links is updated in the congestion information register (CIR) (described in section 4.2). According to Table 2 a link is congested if the value is greater than or equal to 2(binary 10). A congested link is marked as 1 and the non-congested link is marked as 0. According to the proposed algorithm header format is shown in Fig.5. 1-bit *RoutIsSet* and 4bit NoH are added to determine whether paths are set for source routing or not.

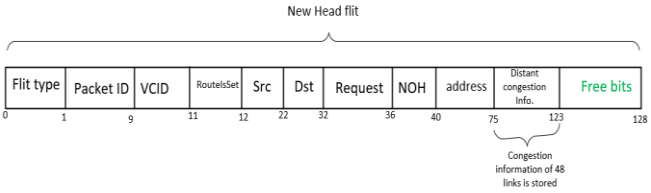


Fig. 5. A new head flit structure for implementation of the proposed algorithm

4.2 Router Architecture for Proposed Algorithm

The Baseline architecture of the router is the same as used in [28], 128bit of NoC, mesh topology, credit-based wormhole flow control, two VCs, and 2-stage Pipeline except those two new modules added to implement a proposed algorithm. Congestion information Register (CIR) module and Path Computation Unit (PCU). This modified architecture of the router is shown in Fig**.**6, CIR stores and updates the congestion information after calculation of remote link in data packets. CIR consist of four registers each of 16bit that holds the congestion information of non-local nodes. Packets coming from the neighbor node

having congestion information of (three Hops away from destination node along north, east, west and south) is updated in these register bank.

PCU computes the path for source routing. Step 2 and Step 3 in algorithm 1 are being executed by this unit. A packet from the CIR module enters in PCU and checks the status bit *RouteIsSet* from the header of the packet. If this bit is set, it executes step 3 i.e., the output node is determined by the LSB value stored in a packet header, shift encoded path bit to left, and decrement to NoH. If the status bit *RouteIsSet* is 0. It calculates Intermediate nodes and checks if the destination node is outside the region of I_x or I_y. It calculates the path as described above in the Algorithm 1 strategy and stores the encoded path in the header of the packet.

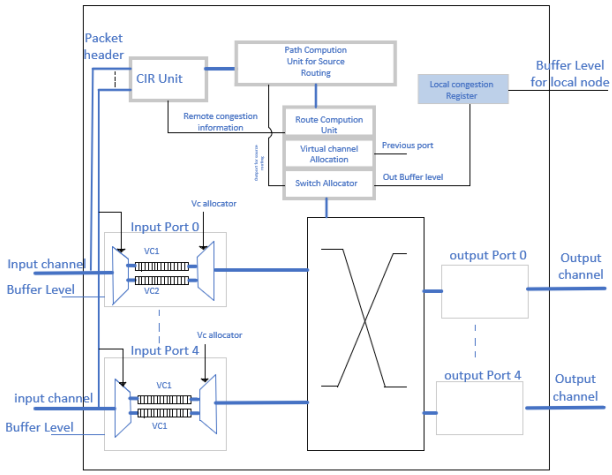


Fig. 6. Router Architecture to implement the proposed algorithm

Above implementation have a negligible effect on latency and area because to implement a CIR unit only 48 registers are required. PCU only performs addition and compare function in addition with requirements of some storage register to store 4bits encoded path. We will evaluate power consumption in the next section.

4.3 Workflow for Implementing Proposed Algorithm

Process flow graph of the proposed algorithms is described by the block diagram shown in Fig. 7. At first, when head flit arrives congestion information of local link is updated to CIR unit as well as the head flit of packet is updated. Rest of congestion information in CIR is also updated by loading congestion from head flit. Head flit contains source and destination address, based on this address intermediate node is calculated. In parallel RCU compute local output node along x-direction or y-direction based on the buffer level in virtual channel. If buffer level is more than prescribed threshold RCU hand over the control to PCU, in PCU algorithm 1 and algorithm 2 work as routing function to compute the complete path to intermediate node and to the destination node. In proceeding subsection explains in detail the working of both algorithm with examples.

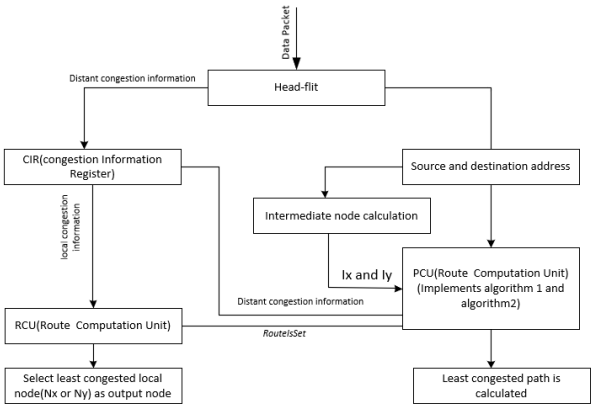


Fig. 7. Flow graph for implementing the proposed routing algorithm.

4.4 An Example for Algorithm 1

An example of proposed algorithm 1 is given in this section. Let us consider a source node is represented by S and a Destination node is represented by D as shown in Fig. 8 at first algorithm checks if congestion in the local neighbor in direction of the destination is less than 10%. Let's assume the buffer level of both neighbor nodes N_x and N_y is greater than 10%. The route computation unit (RCU) will not decide the output direction for the source node. In Parallel the PCU checks if *RoutIsSet* status bit in the packet header and calculate intermediate node I_x and I_y. Let's consider *RoutIsSet* status bit is 0. The algorithm checks destination node is in the region of I_x and I_y. it can be seen in Fig that D is outside of both Intermediate node I_x and I_y. Now the path computation in Step 2 is executed by PCU. Three Paths along x-direction P_{x1} P_{x2}, P_{x3}, and along y-direction P_{y1}, P_{y2} and P_{y3} are compared based on remote congestion information of links from data packets. It is shown in Figure that P_{x1} has minimum congestion i.e. P_{x1}=link1+link2+link6(0+0+0) is 0. The remote congestion address of each link is shown in Fig. So, the Intermediate node along x- direction I_x is selected with path 1 along the x-axis (P_{x1}). Address of path 1 from Table 3 is loaded in the header of the packet as [0001]. Now control goes to Step 3 of Algorithm 1. Here output port is decided based on the LSB of the encoded address stored in the packet header. LSB is 0, which means the output port for the packet is selected along X-axis (East). After shifting 1 bit left the address [0010] and status bit *RoutIsSet* =1 both values are loaded to the packet header and NoH is decremented by one value (NoH=3). For the next output node, Algorithm 1 cycle is repeated (as S and D are in HCA). At first congestion info of the local route is checked. Assume its greater than 10%, RCU will transfer its control to PCU. PCU checks RoutIsSet =1, LSB is 0,

and the output port in the east is selected by decrementing NoH. By Shifting to the left, the new address [0100] is loaded in the packet header again and repeats the cycle. According to LSB =0, the output port is selected along the x-axis to the east. At last, by shifting to the left new address [1000] is loaded in the packet header and repeats the cycle. According to LSB =1, the output port is selected along the y-axis to the north.

For the next cycle, NoH=0 and *RoutIsSet* =0 show packet has been reached the Intermediate node. The complete process is again repeated. Now destination node is inside the region of Intermediate node I_y. The algorithm checks input buffer has a congestion level of less than 50% of Neighbor N_x and N_y. Output with the least congested link is selected. Assume that N_y along y- the axis is selected with the least congested link. As next neighbor node is the destination node(D). So, a packet is again routed to the North N_y neighbor node without comparing the congestion level of buffer along with N_y and N_x.

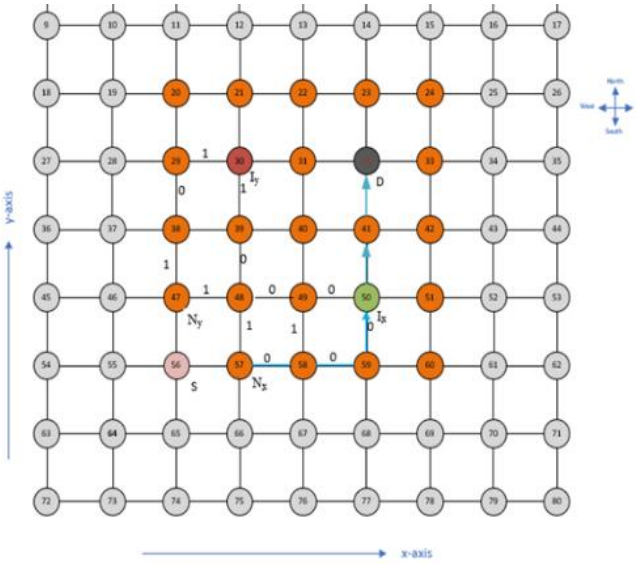


Fig. 8. 2D mesh Network having source and a destination node in HCA to implement Algorithm 1

4.5 An Example for Algorithm 2

Algorithm 2 is Illustrated by another example in this Section. Let's consider source node is in HCA and the destination node is in LCA as shown in Fig.9 Algorithm 2 checks the closeness centrality of an Intermediate node. In this example closeness centrality of I_y consider zero because it lies outside the size of the network. So, packet is routed to low centrality values without knowing local congestion or distant congestion information, in the direction of the destination. A packet is routed along the y-axis to the South. Now both the source node and destination node are in LCA. Distant congestion information could not be found because Neighbor N_y" node is a boundary node. Based on local congestion information packet is routed to neighbor node N_x to the east. Again, because neighbor node N_y is a boundary node. Therefore, based on local congestion information packet is routed to a destination node.

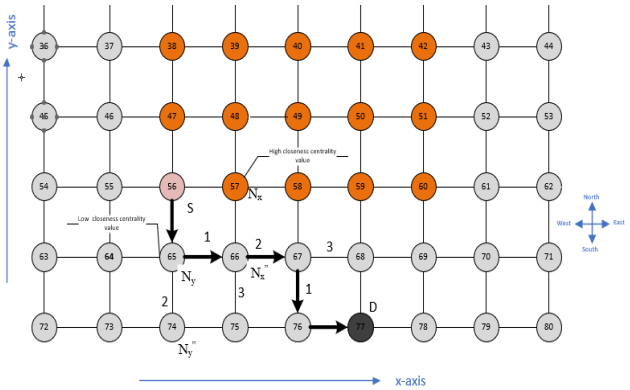


Fig. 9. Example Explaining algorithm 2, Based on local congestion awareness packet is routed to the destination

4.6 Deadlock Freedom

For an adaptive routing algorithm considering deadlock, freedom is Important. The proposed algorithm is a minimal destination-based adaption routing algorithm. which is itself deadlock free. Deadlocks may occur while doing source routing by choosing the path. Router architecture is used in a network having two VCs. VC1 and VC2, VC2 has low priority and is used only by-passing packets when a dead lock occurs. Allocation of VC2 is done only for WS (West South) and WN (West North) turns. So, VC2 is used as an Escape virtual channel [34] when a dead lock occurs.

5. Simulation and Results

Simulation for performance measure of proposed routing algorithm and comparison in term of different measure parameter with other routing algorithms is presented in this section. We need a tool that can simulate the parameters of NOC and choose the ideal parameter for this evaluation. This tool is known as a NOC simulator. Generally, two types of simulators are used named software and hardware simulators. Because of the software simulator's most significant attribute is great robustness. Therefore, for NoC evaluation simulators are usually devolved in the form of software [31]. A software simulator must have specific features such as cyclic accuracy, general for different structures of a network, and notable simulation speed. A highly parameterizable capability of fast modeling of concurrent hardware modules with cyclic level accuracy, a NoCTweak[35] NoC simulator, is proposed in 2012. This simulator has been designed and developed in SystemC with C++ Plugins. For performance evaluation, this simulator provides output results in average latency, average

throughput, and router power (in mW) using a 65nm standard cell CMOS library.

Table 4
Parameter setup for Performance Evaluation

Topology	16x16 2D Mesh
Routing algorithm	Proposed
Switching Technique	Wormhole
No. of Ports	Five
No. of VCs	two
Size of flit	128bits
Length of packet	20flits
Frequency	150 MHz
Feature Size	65nm
Worm-up-time	2000Cycles

NocTweak evaluates the proposed algorithm's performance. Each NoC router has a total of five ports and PEs. Each router port is equipped with two VCs, each with a capacity of sixteen flits. Table 4 contains the specifics of the simulation setup.

5.1 Latency and Throughput Analysis

The average delay of packets generated in the network is the first metric considered in the NoC performance analysis. Traffic patterns and workload must be particular for NoC measurement. The workload dictates how many packets are formed per node on every clock cycle, whereas the traffic pattern determines the destination to the source node on every clock cycle. Using the probability distribution function, if traffic is injected synthetically, these traffic patterns are called synthetic traffic patterns. Mostly used synthetic traffic patterns are uniformly random, Bit-reversal, Transpose, and bit-complement. In this paper, a comparison of a suggested algorithm is done with other three similar algorithms ZARA5-16 [15], Dy-Adaptive [25], and subnet [20] under uniform random, Bit-reversal, and transpose synthetic traffic patterns at different injection rates. For a 2-D mesh 16x16 network, Fig. 9-11 illustrates the performance measure in terms of average latency of the proposed routing algorithm and other three existing routing algorithms under uniform random, Bit-reversal, and transpose traffic patterns. The horizontal axis depicts the injection rate (IR) in flits per node per cycle that defines the workload in a network. The vertical axis represents average latency in cycles.

In uniform, Bit-reversal, and transpose traffics, the proposed method outperforms the other three algorithms, as shown in Fig. 10-12.

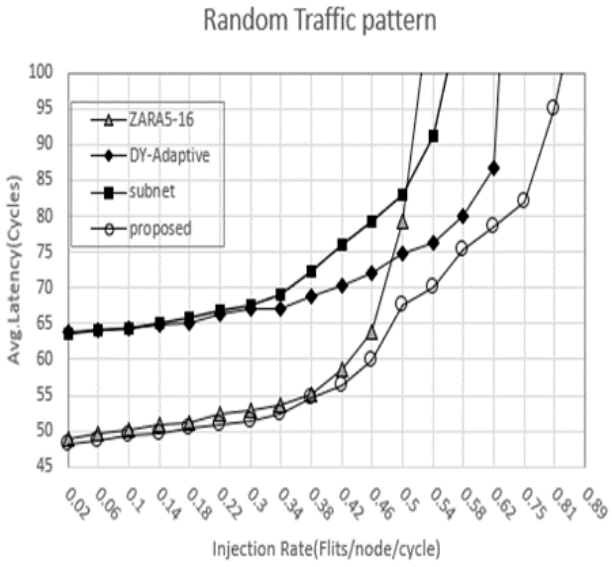


Fig. 10. Average latency evaluation under synthetic uniform random traffic pattern

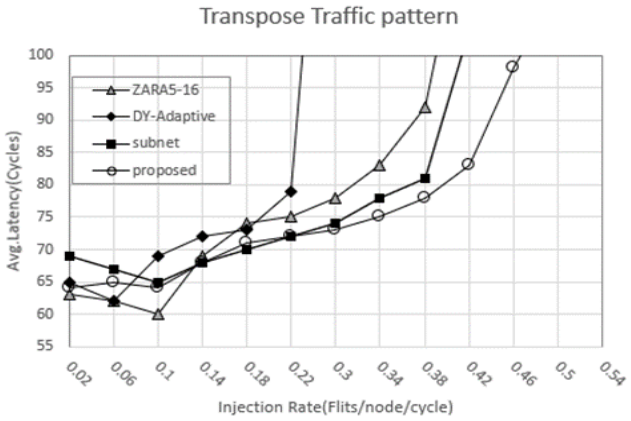


Fig. 11. Average latency evaluation under synthetic Transpose traffic pattern

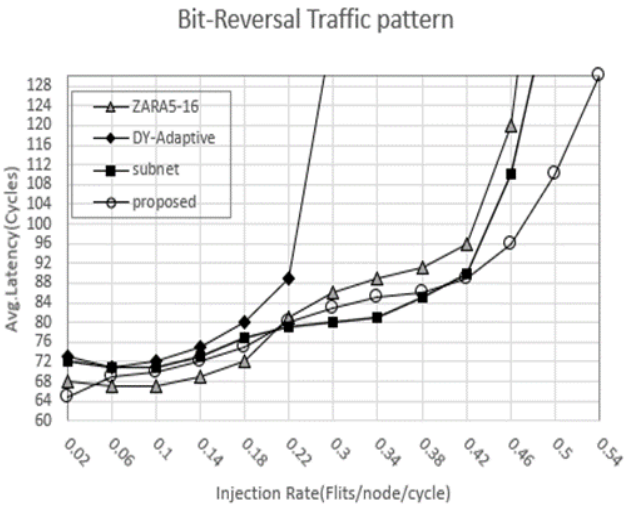


Fig. 12 Average latency evaluation under synthetic Bit-Reversal traffic pattern

According to some authors [31, 25] uniform traffic is unattainable, and the worst outcomes are unimportant in this case. Because of the superior nature of the proposed algorithm, which is minimally adaptable and routes the least crowded way, it outperforms existing algorithms. Fig.10 under

uniform random traffic, which is itself balanced, the proposed algorithm performs well as compared to Dy-Adaptive and ZARA5-16. Because at a high injection rate proposed algorithm route packet to a path having the least congestion. Fig.11 shows average latency under transpose traffic patterns. Since in this traffic packets are usually routed over a long path. Dy-Adaptive is saturated rapidly and in comparison, with the other two algorithms, the proposed algorithm performs better because of load balancing and choosing the least congested non-local route selection. Fig.12 illustrates average latency under bit reversal traffic patterns. Results are similar to the transpose traffic patterns. There is no significant difference and again proposed algorithm outgoes the existing one. In Table 5 it is shown that Average latency of proposed algorithm in comparison with others under various injection rate.

Fig.13-15 illustrates the performance evaluation of the proposed algorithm with the other two algorithms Dy-adaptive and subnet in terms of average throughput. The horizontal axis represents IR in flits per node per cycle and the vertical axis is represented by average throughput in flits per cycle. In Fig. 13 proposed algorithm is compared with the other two algorithms Dy-Adaptive [25] and subnet [20] under uniform traffic patterns. Uniform random traffic is already balanced all over the nodes in the network so there is no significant difference between them.

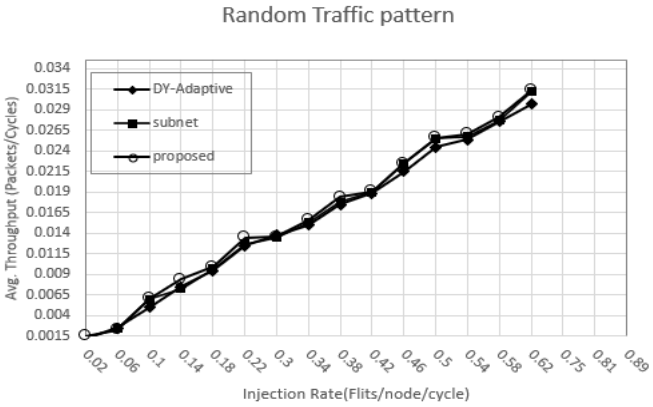


Fig. 13 Average throughput evaluation under synthetic uniform random traffic pattern

In Fig.14, it can be concluded that the proposed algorithm is superior to the Dy-Adaptive routing algorithm under transpose traffic patterns. While comparing subnet, at first proposed algorithm defeats subnet but when IR from 0.38Flits/Node/Cycle increased subnet outgo proposed algorithm. This is because of the highly adaptive nature of the subnet. But at high IR (0.75 Flits/Node/Cycle) proposed algorithm again defeats the subnet and does not degrade soon. Hence proposed algorithm defeats the other two algorithms.

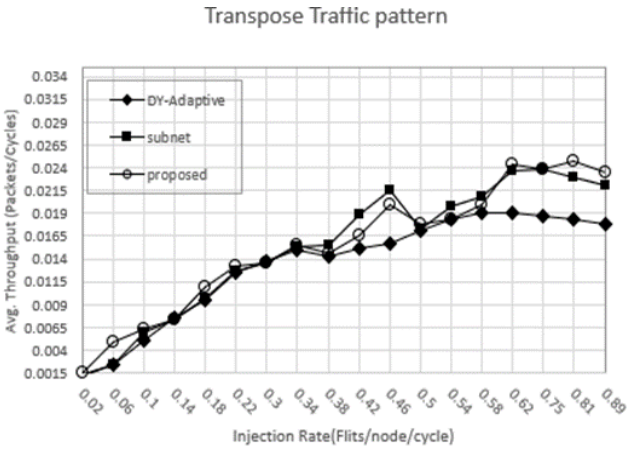


Fig. 14 Average throughput evaluation under synthetic Transpose traffic pattern

In Fig.15 it is noticeable, that our algorithm is improved than the Dy-Aaptive algorithm. However, the proposed algorithm could not beat the subnet by a significant margin. But the results are not the worst and are still competitive and the proposed algorithms are performing well. Table 6. listed the Average Throughput of proposed algorithm in comparison with others under various injection rate.

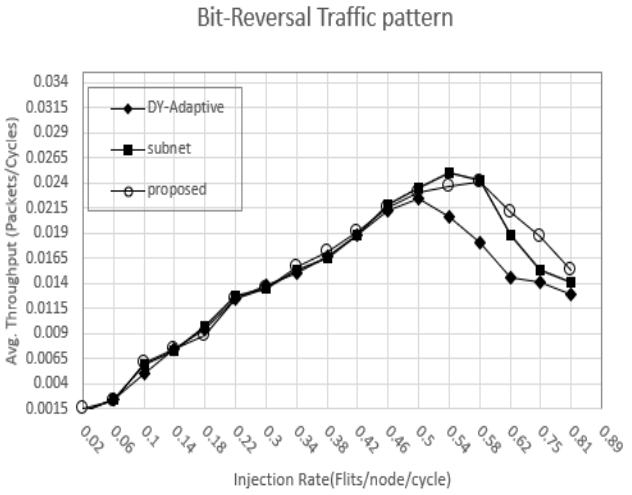


Fig. 15 Average throughput evaluation under synthetic Bit-Reversal traffic pattern

5.2 Power Consumption Analysis

For power evaluation nocTweak [35] simulator utilizes a 65nm standard cell CMOS library. Analysis of power consumption by the router is an important parameter. Since a routing algorithm utilizes a chip as the backbone for its implementation. Implementation of congestion-aware routing causes high power consumption at a high injection rate. A load-balanced congestion-aware routing algorithm distributes the workload all over the network and hence allows efficient utilization of resources and the reduction of workload over a single resource also causes a reduction in power consumption. The proposed routing algorithm does not use an additional network

of wires to detect congestion of non-local nodes. However, it utilizes the CIR module for congestion information propagation via data packet and PCU for computation. Simulation results show that the worst-case scenario of power consumption is not at a high injection rate. Fig.16-17 depicts the evaluation of power consumption of the suggested algorithm in comparison with the Dy-adaptive and subnet routing algorithm.

Fig.16 under a uniform traffic pattern, at a high injection rate Dy-adaptive, performs well as compared to the proposed algorithm. In comparison with the subnet proposed algorithm less power consumption. In Fig. 17 under transpose traffic patterns, power consumption of Dy-Adaptive increases rapidly. The proposed algorithm performs very well at a low injection rate but after injection rate, 0.18flits/node/cycle power consumption is increased and hence subnet has the least power consumption.

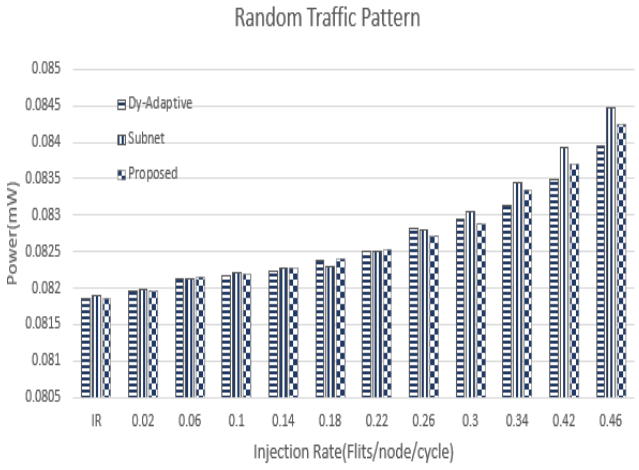


Fig. 16. Power consumption evaluation under synthetic uniform random traffic pattern

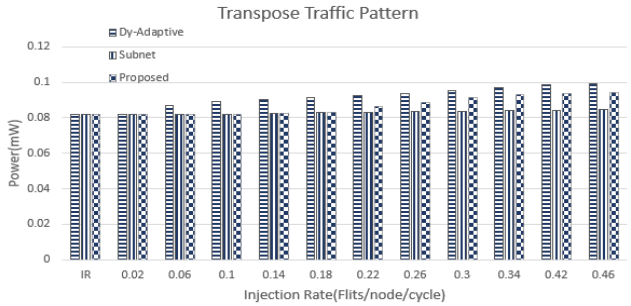


Fig. 17. Power consumption evaluation under Transpose traffic pattern

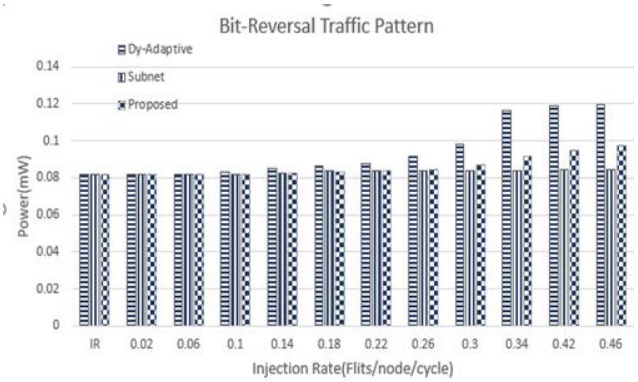


Fig. 18. Power consumption evaluation under synthetic Bit-Reversal traffic pattern

Fig. 18 under bit reversal, the proposed algorithm consumes less energy than a subnet at a lower injection rate but at a higher injection rate subnet has the least power consumption than both Dy-Adaptive and the proposed algorithm. However, under these traffic patterns, it can be concluded that, and no worst scenario makes inefficient design as power consumption is an important parameter in portable NoC Structure.

Table 5

Average latency of proposed algorithm in comparison with others under various injection rate:

Traffic Patterns	Average latency(cycles)			
	Dy-Adaptive	ZARA5-16	Subnet	proposed
Uniform random traffic	68	64	74	64
Transpose Traffic	204	150	120	89
Bit-Reversal Traffic	213	102	100	85

Table 6

Average Throughput of proposed algorithm in comparison with others under various injection rate:

Traffic Patterns	Average Throughput (Packet/Cycle)		
	Dy-Adaptive	Subnet	Proposed
Uniform random traffic	0.015473	0.015847	0.01614
Transpose Traffic	0.013372	0.015255	0.01540
Bit-Reversal Traffic	0.013335	0.014517	0.01479

5.3 Scalability Analysis

Another important parameter to check the efficiency of the algorithm is to check whether it is scalable or not. Since mostly global congestion-aware routing algorithms are not scalable. The scalability of the routing algorithm shows that it can be implemented to

different dimensions of the NoC without performance degradation. Fig. 18 shows average latency under transpose traffic patterns that normalized to the proposed routing algorithm, at different NoC dimensions. When dimensions of NoC increase there is an improvement in average latency. Because at each candidate node proposed algorithm finds the least congested path to the intermediate node and balances load by shifting traffic to the non-congested area. Fig. 19 shows that the proposed algorithm behaves identically to existing algorithms over almost all dimensions of the NoC.

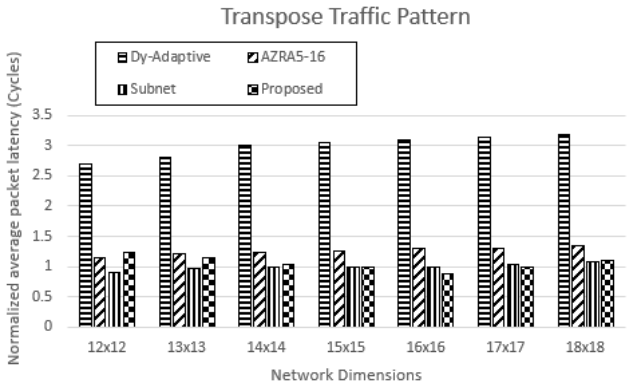


Fig. 19. Comparison of the proposed algorithm with others in terms of normalized average packet latency under synthetic transpose traffic pattern with different network size

6. Conclusion

In the recent era, to process the massive data the number of PEs is increasing on a single chip. Efficient communication between these PEs is a big challenge. One of the best solutions is to design routing algorithm that communicates efficiently between these PEs and routes the packet to the least congested link. In this paper, an improved congestion-aware load balancing algorithm is proposed. The proposed algorithm is a type of minimal adaptive algorithm. The network is divided into two areas, LCA and HCA, based on closeness centrality. For both areas, two algorithms are proposed. Load balancing is achieved by routing the packet from HCA to LCA and hence equally distributed load all over the network. In LCA, a minimal dentation-based adaptive algorithm is implemented by knowing the congestion information of distant nodes. In HCA, an intermediate node is defined in the direction of the destination node, and the least congested path is computed using source routing to this intermediate node by knowing the congestion status of non-local nodes via data packets. To implement the proposed algorithm two additional modules CIR and PCU are included in the baseline router. Simulation results show that the power consumption does not reach to worst-case condition and is still in competition with other comparative algorithms. Moreover, performance evaluation in terms of latency and throughput, shows the prominence of the proposed algorithm more than Dy-Adaptive, AZRA, and subnet. Finally, the proposed algorithm's scalability is proved using various network dimensions. It is shown that average latency and throughput improvement is 31.28% and 5.26% respectively, then existing comparative algorithms.

7. References

[1] J. Duato, S. Yalamanchili, L.M. Ni, "Interconnection networks: an engineering approach", Morgan Kaufmann Publishers, July 2003.

[2] W.J. Dally, B.P. Towles, "Principles and practices of interconnection networks", Elsevier, 2004.

[3] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks", Proceedings of the 38th Design Automation Conference, pp. 684-689, 2001.

[4] ITRS, "International, Technology Roadmap for Semiconductors", Edition Technical Report, 2011.

[5] D. Ayhan, H. Ahangari, and O. Ozturk, "Temperature-aware core mapping for heterogeneous 3D noc design through constraint programming", In 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 312-318. IEEE, 2020.

[6] M. Kumar, V. Laxmi, M.S. Gaur, S.B. Ko, M. Zwolinski, "CARM: congestion adaptive routing method for on chip networks", Proc. of the IEEE International Conference on VLSI Design, pp. 240–245, 2014.

[7] R. Akbar, F. Safaei, S.M.S. Modallalkar, “A novel power efficient adaptive RED-based flow control mechanism for networks-on-chip”, Comput. Electr. Eng, vol. 51, pp. 121–138, April 2016.

[8] N. Wang and P. Valencia, "Traffic allocation: an efficient adaptive network-on-chip routing algorithm design," 2nd IEEE International Conference on Computer and Communications, pp. 2015-2019, 2016.

[9] Y. Ouyang, et al. “A novel low-latency regional fault-aware fault-tolerant routing algorithm for wireless NoC”, IEEE Access 2020, vol. 8, pp.22650–22663, January 2020.

[10] M.K. Khattak, Y. Tang, H. Fahim, E. Rehman, M.F. Majeed, “Effective routing technique: augmenting data center switch fabric performance”, IEEE Access, vol. 8, pp. 37372–37382, February 2020.

[11] F. Rad, et al, “A survey and taxonomy of congestion control mechanisms in wireless network on chip”, Journal of Systems Architecture, vol.108, pp.101807, July 2020.

[12] M. Abdollahi, S. Mohammadi, "Vulnerability assessment of fault-tolerant optical network-on-chips”, Journal of Parallel Distrib. Comput., Vol.145, pp. 140–159, March 2020

[13] T. Bjerregaard, and S. Mahadevan, “A survey of research and practices of Network-on-chip”, ACM Comput. Survay, Vol. 38, pp. 1-es, 2006.

[14] F. Farahnakian, M. Ebrahimi, et al, “Adaptive load balancing in learning-based approaches for many-core embedded systems”, international Journal of Supercomputing. Vol.68 (3), pp. 1214–1234, 2014.

[15] A. Reza, F. Safaei, and E. Khodadad, “A novel adaptive congestion-aware and load-balanced routing algorithm in networks-on-chip”, Journal of Computers and Electrical Engineering , vol. 71 pp. 60-76, 2018.

[16] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, and P. Liljeberg, "Optimized Q-learning model for distributing traffic in on-chip networks", IEEE 3rd International Conference on Networked Embedded Systems for Every Application, pp. 1-8, 2012.

[17] C. Chen, et al, “Link-sharing: regional congestion aware routing in 2D NoC by propagating congestion information on idle links”, IEEE 3rd International Conference on Integrated Circuits and Microsystems, pp. 291-297, November 2018.

[18] L. Shaoli, et al. "FreeRider: non-local adaptive network-on-chip routing with packet-carried propagation of congestion information", IEEE Transactions on Parallel and Distributed Systems, vol. 26, pp. 2272-2285, 2014.

[19] J. Fang, D. Zhang, and X. Li, “ParRouting: an efficient area partition-based congestion-aware routing algorithm for NoCs”, MDPI Journal of Micromachines, vol.11(12), pp.1034, 2020

[20] N. Taherkhani, R. Akbar, F. Safaei, and M. Moudi, "A congestion-aware routing algorithm for mesh-based platform networks-on-chip", Microelectronics Journal , vol. 114, pp. 105145, 2021 .

[21] G. Du, J. He, Y. Song, D. Zhang, and H. Wu, “Comparison of NoC routing algorithms based on packet-circuit switching”, Proc. of the 2013 IEEE Third International Conference on Information Science and Technology, Yangzhou, China, pp.707-710, March 2013

[22] M. Li, Q.-A. Zeng and W.-B. Jone, “DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip”, Proc. of the 43rd Annual Design Automation Conference, pp. 849–852, 2006.

[23] R. Xie, J. Cai, X. Xin and B. Yang, “MCAR: non-local adaptive Network-on-Chip routing with message propagation of congestion information”, Microprocessors and Microsystems journal, vol. 49, pp. 117–126, 2017.

[24] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip", IEEE 14th International Symposium on High Performance Computer Architecture, pp. 203-214, 2008.

[25] J. Hu and R. Marculescu, "DyAD: smart routing for networks-on-chip", Proceedings of the 41st annual Design Automation Conference in IEEE, pp. 260-263, 2004.

[26] P. Lotfi-Kamran et al, "BARP-a dynamic routing protocol for balanced distribution of traffic in NoCs", In IEEE2008 Design, Automation and Test in Europe, pp. 1408-1413, 2008

[27] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, Z. Navabi, “BARP-a dynamic routing protocol for balanced distribution of traffic in NoCs”, In Proceedings of the conference on Design, Automation and Test in Europe, pp. 1408-1413, 2008.

[28] Ma S, Jerger NE, and Wang Z. “DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip categories and subject descriptors”, In Internatinal symposium on computer architecture, pp. 413-424, 2011.

[29] Ebrahimi M et al, “CATRA- congestion aware trapezoid-based routing algorithm for

on-chip networks", IEE Design automation and test in Europe conference and exhibition, pp. 320-325, March 2012.

[30] S. Xu, J. Wu, B. Fu, M. Chen, and L. Zhang, "Efficient regional congestion awareness for load balance with aggregated congestion information.", In IEEE 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 93–99, March 2017.

[31] R. Akbar and F. J. N. C. N. Safaei, "A novel congestion-aware routing algorithm with prediction in mesh-based networks-on-chip", International Nano Communication Networks in Elsevier , vol. 26, p. 100322, 2020.

[32] R.S. Ramanujam, and B. Lin, "Destination-based adaptive routing on 2D mesh networks", In proc. of IEEE Symposium on Architectures for Networking and Communications Systems, pp. 1-12 , Oct 2010.

[33] M. Ramakrishna, P.V. Gratz, and A. Sprintson, "GCA: Global congestion awareness for load balance in networks-on-chip", IEEE/ACM Internatinal Symposium on Networks-on-Chip, vol. 27, pp. 2022-2033, Nov. 2013.

[34] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers", 39th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 333-346, 2006.

[35] A. T. Tran, and B. Baas, "NoCTweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip", VLSI Computation Lab, ECE Department, University of California, 2012.

[36] K. Ahmad et al., "Congestion-aware routing algorithm for noc using data packets", Wireless Communications and Mobile Computing journal, vol. 2021

[37] K. Ahmad et al., "Congestion-aware routing algorithm for noc using data packets", Wireless Communications and Mobile Computing journal, vol. 2021.

